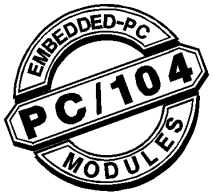


DM5852HR/DM6852HR
Isolated digital I/O-module
User's Manual

Real Time Devices Finland Oy

Hardware release 2.0

Real Time Devices Finland Oy
Lepolantie 14
FIN-00660 Helsinki, Finland
Tel: (+358) 9 346 4538
Fax: (+358) 9 346 4539
Email: sales@rtdfinland.fi
URL: www.rtdfinland.fi

 RTD Finland Oy 1996-2001

IMPORTANT

Although the information contained herein has been carefully verified RTD Finland Oy assumes no responsibility for any errors that may occur, for any damage to property or persons resulting from improper use of this manual or from the related software. RTD Finland Oy also reserves the right to alter the contents of this manual, as well as features and specifications of this product at any time, without prior notice.

Published by
Real Time Devices Finland Oy
Lepolantie 14
FIN-00660 Helsinki, Finland

Copyright © 1996-2001 by RTD Finland Oy
All rights reserved

Printed in Finland

Table of Contents
==========

INTRODUCTION
- **General purpose digital I/O**
- **Isolated power outputs**
- **Isolated optocoupler inputs**
- **Mechanical description**
- **Connector description**
- **What comes with your board**
- **Board accessories**
  - **Application software and drivers**
  - **Hardware accessories**
- **Using this manual**
- **When you need help**

CHAPTER 1 - BOARD SETTINGS
- **Factory-configured jumper settings**
- **Base Address jumpers**
- **Interrupt and Interrupt channels**
- **External Trigger source**
- **Pull-up or Pulldown jumpers**
- **Isolated Input range selection jumpers**

CHAPTER 2 - BOARD INSTALLATION
- **Board installation**
- **External I/O connections**
  - **Isolated output connector**
  - **Isolated input connector**
  - **50-pin RTD Expansion connector**

CHAPTER 3 - HARDWARE DESCRIPTION
- **Digital I/O, 8255 Programmable digital I/O**
- **Isolated transistor outputs**
- **Isolated optocoupler inputs**
- **Interrupts**
- **Triggering**

CHAPTER 4 - BOARD OPERATION AND PROGRAMMING

Defining the I/O map

Base+0	**Output latch A**
Base+1	**Output latch A**
Base+2	**Setup Register**
Base+3	**Clear interrupt**
Base+4	**PPI Port A (Read/Write)**
Base+5	**PPI Port B (Read/Write)**
Base+6	**PPI Port C (Read/Write low byte Read/Write high byte)**
Base+7	**PPI Port CW (Write)**

Programming the DM6852HR

Clearing and setting bits in a port

Initializing the 8255 PPI

Isolated digital output programming

Triggering of isolated digital outputs

Isolated digital input programming

Interrupts

What is an interrupt?

Interrupt request lines

8259 Programmable interrupt controller

Interrupt mask register (IMR)

End-Of-Interrupt (EOI) Command

What exactly happens when an interrupt occurs?

Using interrupts in your program

Writing an interrupt service routine (ISR)

Saving the startup IMR and interrupt vector

Common Interrupt mistakes

APPENDIX A - DM5852HR/DM6852HR Specifications

List of Illustrations

1-1 Board layout showing jumper locations 8
1-2 Base Address jumpers 10
1-3 Pull-up and Pull-down jumpers for Ports A, B and C 11
1-4 Optocoupler input range jumpers 12

2-1 DM6852HR integrated with a PC/104 dataModule stack 15
2-2 19" Eurocard rack installation with an integrated PC/104 dataModule and EUROCARD cpuModule computer system. 16
2-3a **Power output interface** using the spring loaded connector 17
2-3b **Power output interface** using 50-pin header connector 18

2-4a Discrete wire spring loaded connector pinout for **Isolated digital inputs** 19
2-4a Header connector pinout for **Isolated digital inputs** 19
2-5 50-pin expansion interface connector layout 20

3-1 DM6852HR Block Diagram 22

This user's manual describes the operation of the DM6852HR Isolated digital board.

Some of the key properties of the DM5852HR/DM6852HR include:

* 8 MOSFET buffered channel-by-channel isolated digital outputs
* 4 Optocoupled digital inputs with jumper selectable ranges
* 24 non isolated TTL/CMOS compatible 8255-based digital I/O lines
* Optional Pull-up or Pull-down resistors
* Simple I/O or externally triggered output control
* Double buffered outputs with data storage register
* Software selected interrupts
* Support for direct PC/104 interface with RTD dataModules
* Single +5V operation
* **DM5852HR** is XT bus and **DM6852HR** is an AT bus version
* PC/104 form factor

The following paragraphs briefly describe the major features of the DM6852HR. A more detailed discussion is included in Chapter 3 (Hardware description) and in Chapter 4 (Board operation and programming. The board setup is described in Chapter 1 (Board settings).

General purpose digital I/O

The DM6852HR board has 24 8255-based TTL/CMOS compatible digital I/O lines which can be directly interfaced with external devices or signals to sense switch closures, trigger digital events, or activate solid-state relays. These lines may be pulled down or pulled up with a 10Kohm resistor networks. Installation procedures will be described later in this manual.

Isolated power outputs

8 high current MOSFET buffered outputs may be used to directly drive heavy loads such as solenoids, relays, motors, actuators, heaters etc. The output transistors are driven with an advanced FET-driver that protects the output transistor. Optocouplers are used to isolate the high current output channels from the computer. The high power outputs are controlled with a double buffered latch structure. An external trigger may be used to transfer the preloaded data from the first storage latch to the second output latch without the attention of the host computer. An interrupt may be asserted after the trigger condition has ocurred.

Isolated optocoupler inputs

4 optocoupler inputs may be used to connect high voltage signals to your system. Four channel-by-channel jumper configurable input ranges are available: +5V, +12V and +24V. The optocoupler inputs have a reverse voltage protection diode across the input. This enables AC-signals to be connected to the inputs. The input diode will act as a rectifier.

Mechanical description

The DM6852HR is designed on a PC/104 form factor. An easy mechanical interface to both PC/104 and EUROCARD systems can be achieved. Stack your PC/104 compatible computer directly on the DM6852HR using the onboard mounting holes. **DM5852HR** is XT bus and **DM6852** is an AT bus version. You may use the DM6852HR board in an IDAN system in a single high or alternatively a double high frame.

Connector description

There is a 50 pin digital interface connectors on the DM6852HR to directly interface to the non isolated 8255-based digital I/O signals. The signal definition of this connector is compatible with the digital PC/104 dataModules manufactured by RTD.

Isolated outputs and inputs are connected to the DM6852HR by either a terminal block discrete wire connector or with a 50-pin flat ribbon cable header connector. Use this type of interface connector with a TB50 screw terminal block. Please consult the factory for more details on different connector options.

What comes with your board

You receive the following items in your DM6852HR package:

* DM6852HR Isolated digital interface module
* User's manual

Note: Software and and drivers are available on our website.

If any item is missing or damaged, please call Real Time Devices Finland Customer service department at (+358) 9 346 4538.

Board accessories

In addition to the items included in your DM6852HR delivery several software and hardware accessories are available. Call your distributor for more information on these accessories and for help in choosing the best items to support your instrumentation system.

* Application software and drivers

* Hardware accessories

The DM6852HR module is available in the ultra compact robust Aluminium enclosure system IDAN. For more information please visit our website at www.rtdfinland.fi.

Real Time Devices can supply a complete set of accessories for your DM6852HR card. These include PC/104 power supplies, Terminal boards (TB50) and other interconnection systems.

Using this manual

This manual is intended to help you install your new DM6852HR/DM6852 card and get it running quickly, while also providing enough detail about the board and its functions so that you can enjoy maximum use of its features even in all applications.

When you need help

This manual and all the example programs will provide you with enough information to fully utilize all the features on this board. If you have any problems installing or using this board, contact our Technical Support Department (+358) 9 346 4538 during European business hours, or send a FAX to (+358) 9 346 4539 or Email to sales@rtdscandinavia.fi. When sending a FAX or Email request, please include your company's name and address, your name, your telephone number, and a brief description of the problem.

CHAPTER 1 - BOARD SETTINGS

The DM6852HR Isolated digital I/O board has jumper settings you can change to suit your application and sensor input configuration. The DM6852HR is factory configured with a +5V input range configuration. The factory settings are listed and shown in the diagram in the beginning of this chapter.

Factory-configured jumper settings

Table 1-1 illustrates the factory jumper setting for the DM6852HR . Figure 1-1 shows the board layout of the DM6852HR and the locations of the jumpers. The following paragraphs explain how to change the factory jumper settings to suit your specific application.

Table 1-1 - Factory jumper settings (see figure 1-1 for detailed locations)

JUMPER NAME	DESCRIPTION OF JUMPER	NUMBER OF JUMPERS	FACTORY SETTING JUMPERS INSTALLED
BASE ADDRESS	BASE ADDRESS	5	300h INSTALLED
PA,PB,PC	PU/PD FOR PPI PORTS	3-Terminal	PULLDOWN
IN1, IN2, IN3, IN4	INPUT RANGE FOR ISOLATED INPUTS 1-4	3	+5V RANGE

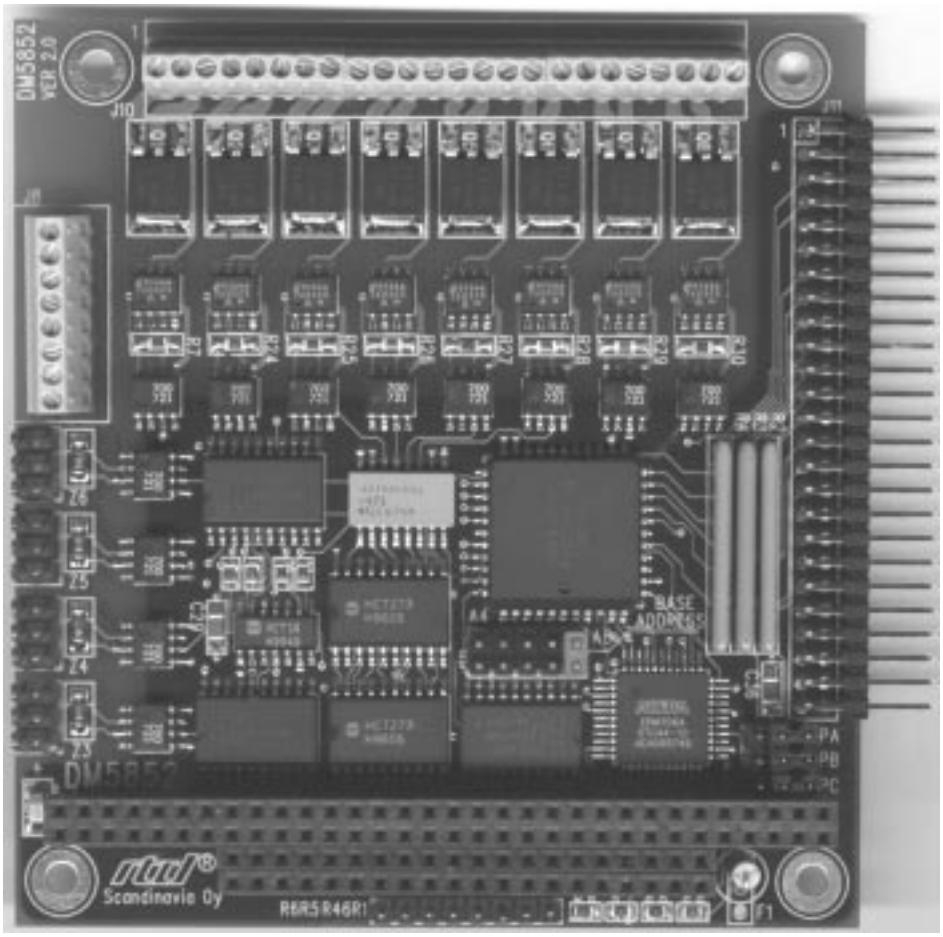


Figure 1-1 - Board layout showing jumper locations

Base Address jumpers (Factory setting: 300h)

The most common cause of failure when you are first setting up your module is address contention. Some of your computers I/O space is already occupied by other internal I/O devices and expansion boards: When the DM6852HR attempts to use its reserved I/O addresses already in use by another peripheral device , erratic performance may occur and data read from the board may be corrupted.

To avoid this problem make sure you set up the base address first using the five jumpers marked "BASE ADDRESS" which let you choose from 32 different I/O addresses in your computers I/O map. Should the factory installed setting of 300h be incompatible to your system configuration, it may be changed to another using the options as illustrated in Table 1-2. This table shows the switch settings and their corresponding values in hexadecimal values. Make sure that you verify the correct location of the base address jumpers. When the jumper is removed it corresponds to a logical "0", connecting the jumper to a "1". When you set the base address of the module, record the setting in the back cover of this manual after the Appendices.

Table 1-2 Base Address jumper settings

Base Address jumper settings DM6852HR			
Base Address Hex / (Decimal)	Jumper Settings Addr A4 A5 A6 A7 A8	Base Address Hex / (Decimal)	Jumper Settings A4 A5 A6 A7 A8
200 / (512)	0 0 0 0 0	300 / (768)	0 0 0 0 1
210 / (528)	1 0 0 0 0	310 / (784)	1 0 0 0 1
220 / (544)	0 1 0 0 0	320 / (800)	0 1 0 0 1
230 / (560)	1 1 0 0 0	330 / (816)	1 1 0 0 1
240 / (576)	0 0 1 0 0	340 / (832)	0 0 1 0 1
250 / (592)	1 0 1 0 0	350 / (848)	1 0 1 0 1
260 / (608)	0 1 1 0 0	360 / (864)	0 1 1 0 1
270 / (624)	1 1 1 0 0	370 / (880)	1 1 1 0 1
280 / (640)	0 0 0 1 0	380 / (896)	0 0 0 1 1
290 / (656)	1 0 0 1 0	390 / (912)	1 0 0 1 1
2A0 / (672)	0 1 0 1 0	3A0 / (928)	0 1 0 1 1
2B0 / (688)	1 1 0 1 0	3B0 / (944)	1 1 0 1 1
2C0 / (704)	0 0 1 1 0	3C0 / (960)	0 0 1 1 1
2D0 / (720)	1 0 1 1 0	3D0 / (976)	1 0 1 1 1
2E0 / (736)	0 1 1 1 0	3E0 / (992)	0 1 1 1 1
2F0 / (752)	1 1 1 1 0	3F0 / (1008)	1 1 1 1 1
0 = NOT JUMPERED , 1 = JUMPER INSTALLED			

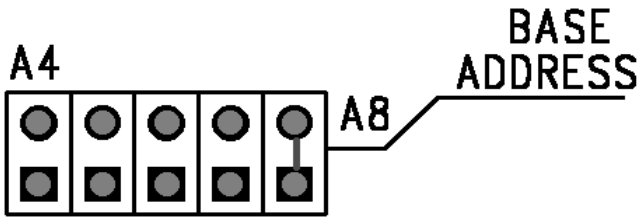


Fig. 1-2 Base Address jumpers set to Base Address 300H

Pull-up or Pull-down resistor selection for digital I/O PA, PB, PC
(Factory setting: PA, PB and PC pulled down)

The 8255 programmable digital I/O interface provides 24 TTL/CMOS compatible lines which can be interfaced with external devices. These lines are divided into three groups: eight Port A lines, eight Port B lines and eight Port C lines. You can connect pull-up or pull-down resistor networks for ports A, B and C. You may want to pull lines up for connection with switches. This will pull the lines high if the switch is disconnected. Or you may want to pull lines down for connection to relays which control turning motors on or off. The port A, B and C lines of the 8255 programmable digital I/O interface are set as inputs after reset. This can cause the external devices connected to these lines to operate erratically. Pulling these lines down, when the board is powered up the lines will not be active before the 8255 is initialized. Figure 1-3 shows both Ports A, B and C set in the Pull-down configuration.

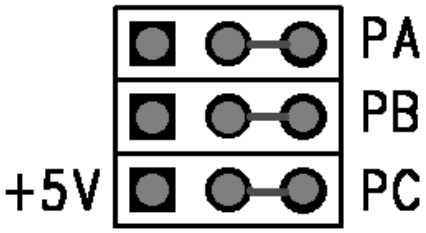


Fig. 1-3 Pull-up and Pull-down jumpers for Ports A, B and C.

PC4-PC7 Optocoupler isolated inputs (Factory setting: +5V range)

The Optocoupler inputs can be jumper configured for different input ranges. Each channel can be set for a different range. The different ranges use different current limiting resistors at the optocoupler diode input. Make sure you set the range correctly since overloading the optocoupler with overcurrent may cause permanent damage to the device. All the channels are set in a similar manner. Figure 1-4 illustrates one channel set for a TTL-level input signal.

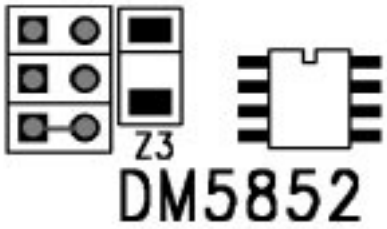


Fig. 1-4 Optocoupler input range jumper set to +5V input range

CHAPTER 2 - BOARD INSTALLATION

The DM6852HR Isolated digital interface board is very easy to connect to your control system. Direct interface to PC/104 systems as well as EUROCARD boards is provided. This chapter tells you step-by-step how to install your DM6852HR in your system.

After completing the installation use the diagnostic software to fully verify that your board is working.

Board Installation

Keep your board in the antistatic bag provided until you are ready to install it into your system. When removing it from the bag, hold the board at the edges and do not touch the components or connectors. Please handle the board in an antistatic environment and use a **grounded** workbench for testing and handling of your hardware.

Before installing the board into your computer, check the jumper settings. Chapter 1 reviews the factory settings and how to change them. If you need to change any settings, refer to the appropriate instructions in Chapter 1. Note that incompatible jumper settings can result in unpredictable board operation and erratic response.

General installation guidelines:

1. Switch the power to your computer OFF
2. Touch the grounded metal housing of your computer to discharge any antistatic build-up and then remove the board from its antistatic bag.
3. Hold the board by the edges and install it in an enclosure . Alternatively , place it on the table on an antistatic surface.
4. Connect the board to the data acquisition board using the twisted pain 50-pin flat cable. Make sure that the orientation of the cable is correct.

Installation integrated with a PC/104 module stack:

* Secure the four PC/104 installation holes with standoffs.

* Connect the 50-pin expansion connector to your non-isolated digital signals and directly connect your isolated input and output control devices to the onboard terminal block connectors.

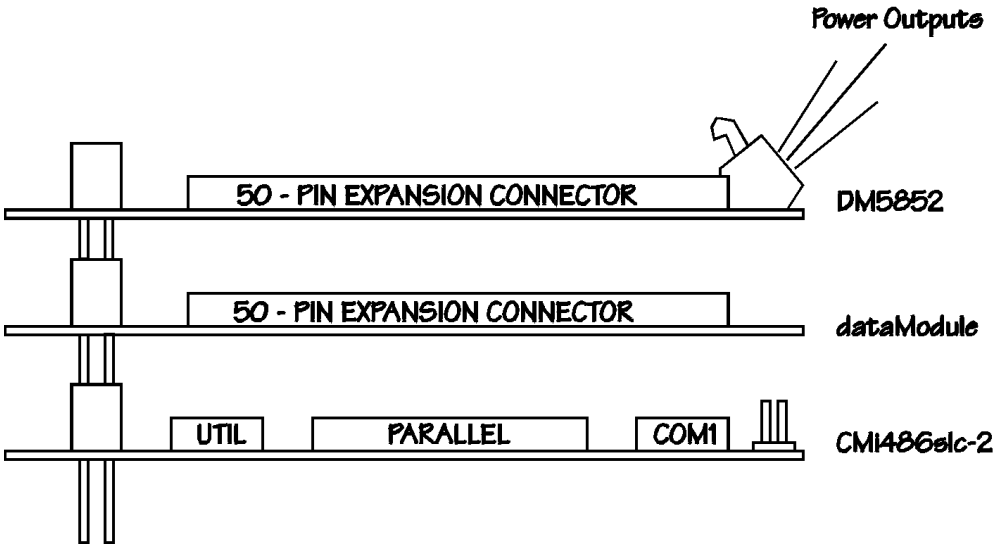


Fig. 2-1 DM6852HR integrated with a RTD PC/104 dataModule stack.

3U rack or enclosure installation with a EUROCARD CPU with one DM6852HR.

* The PC/104 system can easily be inserted into a 19" rack installation using the CPU as a "form factor adapter". Assemble your PC/104 data modules on a RTD single board EUROCARD computer and install the system in a 19" enclosure. Multiple DM6852HR boards can easily be connected to this system. See figure 2-2.

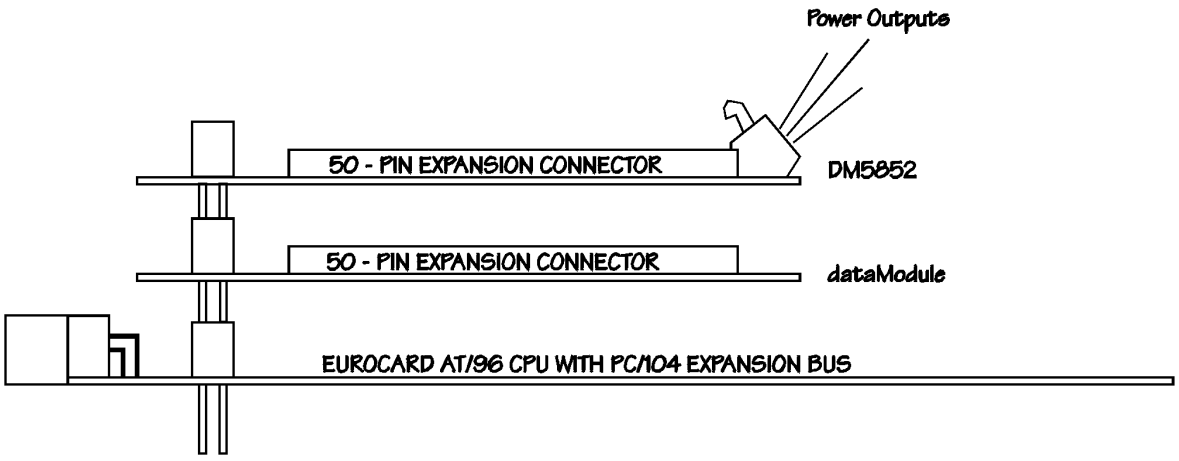


Fig. 2-2 19" Eurocard rack installation with an integrated PC/104 dataModule and EUROCARD cpuModule computer system.

External I/O connections

Figure 2-3 shows the sensor interface connector layout of the DM6852HR. This connector is located toward the top of the board. Refer to this diagram when making isolated signal output connections. Figure 2-4 shows the optoisolated input connector pinout. These connectors are galvanically isolated from the rest of the system.

Figure 2-3a shows the **output** screw terminal discrete wire connector pinout and figure, 2-3b shows the 50-pin flat cable header connector option pinout.

Pin 1 is to the left side of the connector. ALL SIGNALS ARE ISOLATED

#1 SOURCE 1 (gnd)
#2 DRAIN 1
#3 +VS 1 (supply)

#4 SOURCE 2 (gnd)
#5 DRAIN 2
#6 +VS 2 (supply)

#7 SOURCE 3 (gnd)
#8 DRAIN 3
#9 +VS 3 (supply)

#10 SOURCE 4 (gnd)
#11 DRAIN 4
#12 +VS 4 (supply)

#13 SOURCE 5 (gnd)
#14 DRAIN 5
#15 +VS 5 (supply)

#16 SOURCE 6 (gnd)
#17 DRAIN 6
#18 +VS 6 (supply)

#19 SOURCE 7 (gnd)
#20 DRAIN 7
#21 +VS 7 (supply)

#22 SOURCE 8 (gnd)
#23 DRAIN 8
#24 +VS 8 (supply)

Fig. 2-3a **Power output interface** using the terminal block connectors

2-3b shows the 50-pin flat cable header connector option pinout for the isolated outputs.

Pin 1 is to the left side of the connector. ALL SIGNALS ARE ISOLATED

#1,#2	SOURCE 1 (gnd)
#3,#4	DRAIN 1
#5,#6	+VS 1 (supply)
#7,#8	SOURCE 2 (gnd)
#9,#10	DRAIN 2
#11,#12	+VS 2 (supply)
#13,#14	SOURCE 3 (gnd)
#15,#16	DRAIN 3
#17,#18	+VS 3 (supply)
#19,#20	SOURCE 4 (gnd)
#21,#22	DRAIN 4
#23,#24	+VS 4 (supply)
#25,#26	SOURCE 5 (gnd)
#27,#28	DRAIN 5
#29,#30	+VS 5 (supply)
#31,#32	SOURCE 6 (gnd)
#33,#34	DRAIN 6
#35,#36	+VS 6 (supply)
#37,#38	SOURCE 7 (gnd)
#39,#40	DRAIN 7
#41,#42	+VS 7 (supply)
#43,#44	SOURCE 8 (gnd)
#45,#46	DRAIN 8
#47,#48	+VS 8 (supply)

Fig. 2-3b **Power output interface** using 50-pin header connector

Figure 2-4 shows the optoisolated input connector pinout. These connectors are galvanically isolated from the rest of the sytem. 2-4a illustrates the discrete screw terminal connector pinouts, 2-4b illustrates the pinouts for a header connector option.

Pin 1 is to the PC/104 bus side of the connector. ALL SIGNALS ARE ISOLATED

#1	IN+	1	Channel 1
#2	IN-	1 (gnd)	
#3	IN+	2	Channel 2
#4	IN-	2 (gnd)	
#5	IN+	3	Channel 3
#6	IN-	3 (gnd)	
#7	IN+	4	Channel 4
#8	IN-	4 (gnd)	

Fig 2-4a Discrete wire terminal block connector pinout for **Isolated digital inputs**

Pin 1 is to the PC/104 bus side of the connector. ALL SIGNALS ARE ISOLATED

#1,#2	IN+	1	Channel 1
#3,#4	IN-	1 (gnd)	
#5,#6	IN+	2	Channel 2
#7,#8	IN-	2 (gnd)	
#9,#10	IN+	3	Channel 3
#11,#12	IN-	3 (gnd)	
#13,#14	IN+	4	Channel 4
#15,#16	N-	4 (gnd)	

Fig 2-4a Header connector pinout for **Isolated digital inputs**

Figure 2-5 shows the 50-pin expansion interface connector layout of the DM6852HR. This connector is used to interface to the non-isolated general purpose digital I/O signals.

Pin 1 is away from the PC/104 connector.

PC0	*1	*2	EXT_TRIGGER (Input)
PC1	*3	*4	DGND
PC2	*5	*6	DGND
PC3	*7	*8	DGND
PC4	*9	*10	DGND
PC5	*11	*12	DGND
PC6	*13	*14	DGND
PC7	*15	*16	DGND
PB0	*17	*18	DGND
PB1	*19	*20	DGND
PB2	*21	*22	DGND
PB3	*23	*24	DGND
PB4	*25	*26	DGND
PB5	*27	*28	DGND
PB6	*29	*30	DGND
PB7	*31	*32	DGND
PA0	*33	*34	DGND
PA1	*35	*36	DGND
PA2	*37	*38	DGND
PA3	*39	*40	DGND
PA4	*41	*42	DGND
PA5	*43	*44	DGND
PA6	*45	*46	DGND
PA7	*47	*48	DGND
+5V (fused)	*49	*50	DGND

Fig. 2-5 50-pin expansion interface connector layout

CHAPTER 3 - HARDWARE DESCRIPTION

Chapter 3 - **Hardware description** describes the major features of the DM6852HR: the 8255 based digital I/O, Isolated power transistor outputs, isolated optocoupler inputs, interrupts and triggering.

Figure 3-1 shows the general block diagram of the DM6852HR. This chapter describes the major features of the DM6852HR: the 8255 based digital I/O, isolated power transistor outputs, isolated optocoupler inputs, interrupts and triggering.

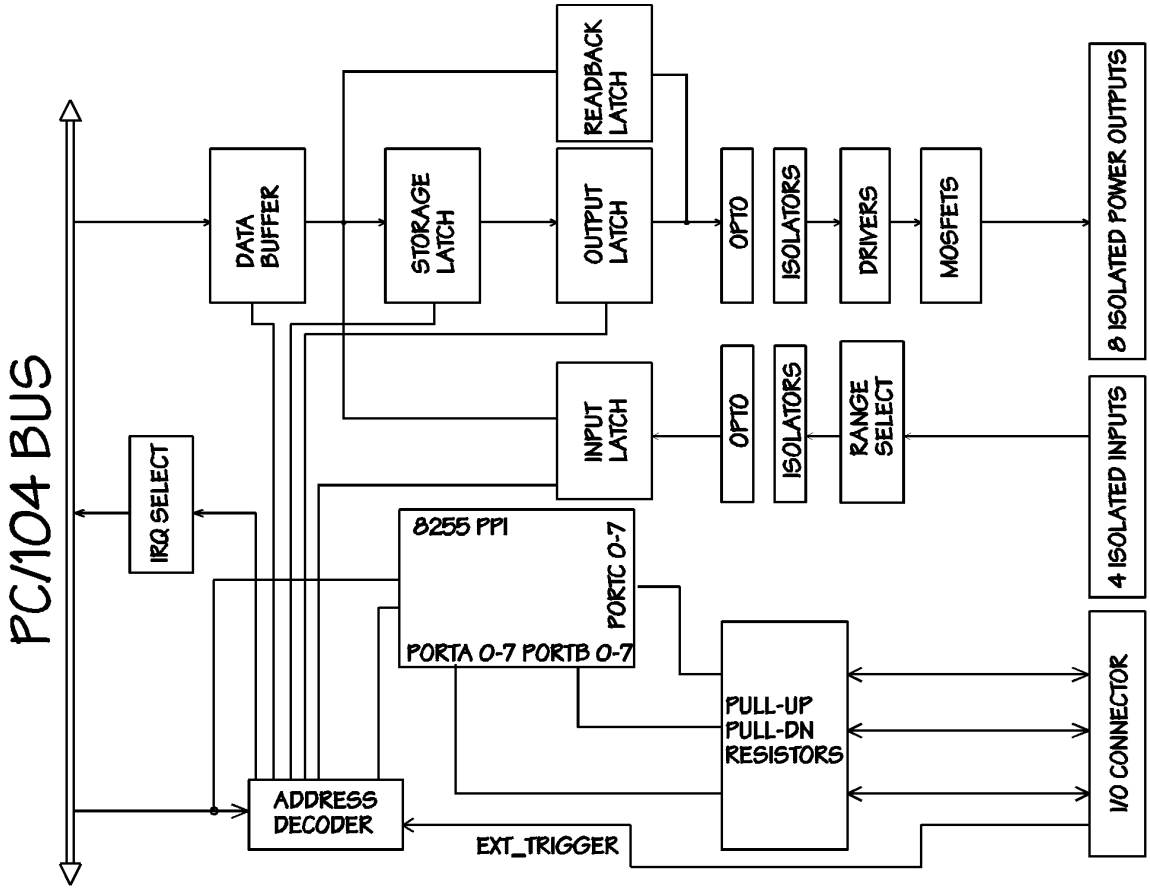


Fig. 3-1 DM6852HR Block diagram

Digital I/O, Programmable Peripheral Interface (PPI)

The programmable peripheral interface (PPI) is used for digital I/O functions. This high performance TTL/CMOS compatible chip has 24 digital I/O lines divided into two groups of 12 lines each.

Group A: Port A (8) lines and Port C upper (4) lines
Group B: Port B (8) lines and Port C lower (4) lines

Port A, Port B and Port C are available at the 50 pin expansion connector.You can use Ports A , B and C in one of these three operating modes:

Mode 0: Basic I/O. Lets you use simple input output functions for a port. Data is written to or read from the specified port

Mode 1: Strobed Input/Output. Lets you transfer data I/O from port A in conjunction with strobed or handshake signals.

Mode 2: Strobed bi-directional input/output. Lets you communicate with an external device through port A. Handshaking is similar to mode 1.

Available Port direction definitions:

Port A may be Inputs or Outputs
Port C lower bits may be Inputs or Outputs
Port C higher bits may be Inputs or Outputs

All these modes are discussed in detail in the 8255 datasheet available from Intel.

Isolated outputs

The Isolated output stage of the DM6852HR consists of 5 major parts:

1. Data storage latch
2. Output latch
3. Optocouplers
4. N-channel MOSFETS

1. Data storage latch

The "data storage latch" is cleared after reset. The purpose of the data storage latch is to store the next output control pattern to the output power MOSFET's. The storage latch is loaded by performing an 8-bit write to the address of the "data storage latch". A software write to the "output latch" or an external trigger transfers the stored pattern to the output FET's. The "data storage latch" is located in address BASE+0. Data cannot be read back from this register. It is possible to prevent the system reset from clearing this register by setting a bit in the board status register. This will prevent accidental state change of the outputs if a system reset occurs due to a watchdog timer reset.

2. Output latch

The "output latch" is also cleared after reset. This ensures that the power output transistors are "OFF" or "non-conductive" after reset and in a predictible state. The purpose of the "output latch" is to store the control output data to the output power MOSFET's. Data can be written to the "storage latch" without affecting the outputs. A software writes the address of the "output latch" (BASE+1) and will transfer the contents of the "storage latch" to the "output latch" and the power outputs. It is also possible to do this transfer with an external trigger. When using an external trigger you can change to the next output pattern without involving the host CPU. After such an event an interrupt to the host can be asserted to signal to the host that the data Storage latch must be updated for the next state. This provides deterministic control to the outputs. The interrupt latency time uncertainty will not affect the response time of the outputs. Data can be read back from this register from address BASE+1. It is possible to prevent the system reset from clearing this register by setting a bit in the board status register. This will prevent accidental state change of the outputs if a system reset occurs.

3. Optocouplers

Small SMD-optocouplers are used to isolate each output channel from the computer. Individual optocouplers are used for each channel. The optocouplers are directly connected to the "output latch". The optocouplers are connected in a non-inverting configuration.

4. N-channel MOSFETs

The output drive device is an N-channel MOSFET. It is connected in a configuration that sinks current. This means that current will flow through the drain to the source when turned "on". The output transistor can be used as an inverting level shifter if you connect a resistor to the supply input voltage of the channel.

Isolated inputs

The Isolated input stage of the DM6852HR consists of 3 major parts:

1. Level Selection jumpers
2. Optocouplers
3. Schimdt trigger buffers.

1. Level selection jumpers

Three preset voltage input levels can be selected using onboard jumpers. These include: +5V (TTL) , +12V (Automotive) and +24V (Automotive /Industrial). Four groups of three jumpers are used for the range selection for all channels. The top-most is used for input channel 1, the bottom-most (next to the PC/104 bus connector) for input channel 4. Each channel has three jumper locations. The topmost is +24V, the middle one +12V and the bottom one selects +5V. A reverse voltage protection diode at each input protects the optocoupler from reverse voltages.

2. Optocouplers

Small SMD-optocouplers are used to isolate each channel from the computer. Individual optocouplers are used for each channel. The optocouplers are directly connected to the "output latch". The optocouplers are connected in a non inverting configuration.

3. Schmidt triggers

The output of the optocoupler is connected to a buffer to condition the output of the optocouplers, this will prevent any false triggering of the isolated input signals.

Interrupts

The DM6852HR can generate to indicate that an external trigger has occurred. This interrupt can be used to inform the host computer that the "data storage latch" has been transferred to the outputs by an external trigger event. Alternatively it can be used just as an external interrupt input. *Chapter 4* will provide more programming information on how to use the interrupt features of the board.

The interrupt request line is software selectable by programming the MASK register of the interrupt request lines on the DM6852HR. Available interrupts include:

DM5852HR IRQ 5, 6 and 7.
DM6852HR IRQ 5, 6, 7,10, 11 and 12

CHAPTER 4 - BOARD OPERATION AND PROGRAMMING

This chapter shows you how to program and use your DM6852HR. It provides a complete detailed description of the I/O-map and a detailed discussion of programming operations to aid you in integrating this board into your system.

Defining the I/O Map

The I/O map of the DM6852HR is shown in Table 4-1 below. As shown the module occupies 8 addresses. The Base Address (designated as BA) can be set using the jumpers as described in Chapter 1, *Module Settings*. The following sections describe the register contents of each address used in the I/O map.

Table 4-1 DM6852HR I/O Map

Table 4-1 DM6852HR/DM6852 I/O Map			
Register Description	**Read Function**	**Write Function**	**Address in HEX**
Data Storage latch	Digital Inputs 0-3	Writes to Storage latch	BA+0
Output latch	Output Latch data	Writes to Output latch	BA+1
Setup register	Read setup register	Writes to Setup register	BA+2
Clear Interrupt	Reserved	Clear HW interrupt	BA+3
PPI Port A	Reads Port A data	Writes Port A data	BA+4
PPI Port B	Reads Port B data	Writes Port B data	BA+5
PPI Port C	Reads Port C data	Writes Port C data	BA+6
PPI Control Byte	Reserved	Writes to Control byte	BA+7
IRQ_SEL_REG1	Read IRQ mask bits	Write IRQ mask bits	BA+8
IRQ_SEL_REG1	Read IRQ mask bits	Write IRQ mask bits	BA+9
*** BA = Base Address**			

BA+0 Data storage latch **(Write only)** 0x00H after reset

The "data storage latch" stores the next output data to be sent to the "output latch". This data is latched by performing an 8-bit write to **BA+0**.

BA+0 Digital inputs **(Read)**

The optoisolated digital input bits available in connector J8 can be read from address BA+0. Only bits 0-3 are defined.

BA+1 Data output latch **(Write)** 0x00H after reset

The "data output latch" controls the output MOSFETS. Data is transferred from the "data storage latch" to the outputs by performing an 8-bit write to **BA+1**. Direct transfer of data to the output latch can be performed by executing a 16-bit write to BA+0. The address decoder of the board will automatically write consecutive addresses BA+0 and BA+1 with correct data.

BA+1 Data output latch (Read)

Performing a read to the "data output latch" address will return the status of the "data output latch" reflecting the current state of the outputs of the MOSFETS.

BA+2 Setup register (Read/Write) 0x00H after reset

The setup register stores the hardware dependent control bits for selecting different modes of operation for the DM6852HR. The contents of the setup register can be read back from the same address BA+2.

BIT 0

D0 = 1 -> Interrupts are enabled. A rising or falling edge (Depending on the trigger polarity bit) will cause an interrupt and the interrupt status bit is set.
D0 = 0 -> Interrupts are disabled.

BIT 1

D1 = 1 -> "External trigger" is enabled. A rising or falling edge, depending on the trigger polarity bit, will clock the "output data latch". This trigger will transfer the contents of the "storage latch" to the "output latch ".
Note: Software triggering is now disabled!
D1 = 0 -> "External trigger" is disabled. The "output latch" is updated by software

BIT 2

D2 = 1 External Trigger signal is non inverted; rising edge will trigger.
D2 = 0 External Trigger signal is inverted; falling edge will trigger.

BIT 3

D3 = 1 System reset clear Data and Output Latch disabled
D3 = 0 System reset clear Data and Output Latch enabled

BA+3 Clear interrupt (Write) 0x00H after reset

Writing to this address will clear the hardware interrupt line from the DM6852HR to the host.

BA+4 PPI Port A (Read/Write)

Transfers the 8-bit Port A digital input and output data between the module and an external device. A read transfers data from the external device, through connector J11, and into port A; a write transfers the written data from port A through J11 to the external devices.

BA+5 PPI Port B (Read/Write)

Transfers the 8-bit Port B digital input and output data between the module and an external device. A read transfers data from the external device, through connector J11, and into port B; a write transferrs the written data from port B throgh J11 to the external devices.

BA+6 PPI Port C (Read/Write)

Transfers the 4-bit low nibble of the Port C digital input and output data between the module and an external device. A read transfers data from the external device, through connector J11, and into bits 0-7 of port C; a write transfers the written data bits 0-7 from port C through J11 to the external devices.

BA+7 PPI Control byte (Write only)

When bit 7 is set to 1, a write programs the PPI configuration.

D7	1 = Active	Mode Set Flag
D6	**D5**	Mode select for group #1
0	0	Mode 0
0	1	Mode 1
1	1	Mode 2

Group #1

D4	0 = A Output 1 = A Input	Direction of Port A
D3	0 = C upper Output 1 = C upper Input	Direction of Port C Upper (bits 4-7)
D2		Mode select for group #2

Group #2

D1	0 = B Output 1 = B Input	Direction of Port B
D0	0 = C lower Output 1 = C lower Input	Direction of Port C Lower (bits 0-3)

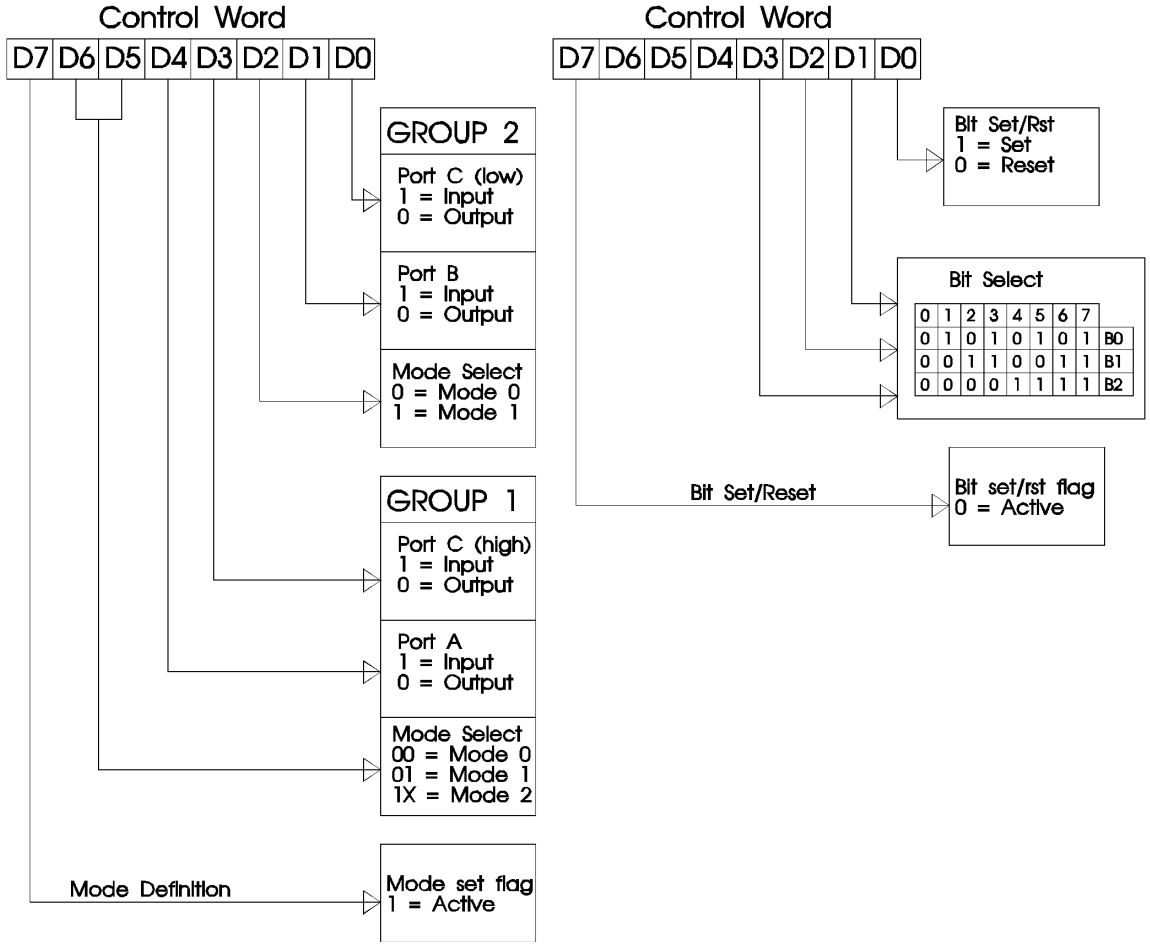
When bit 7 is set to 0, a write can program individual lines of port C.

D7	0 = Active	Set Reset Function bit

D3	**D2**	**D1**	Bit Select
0	**0**	**0**	**PC0**
0	**0**	**1**	**PC1**
0	**1**	**0**	**PC2**
0	**1**	**1**	**PC3**
1	**0**	**0**	**PC4**
1	**0**	**1**	**PC5**
1	**1**	**0**	**PC6**
1	**1**	**1**	**PC7**

D0	1 = Set bit to 1 0 = Set bit to 0

8255 Mode Definition, Bit Set/Reset Feature



BA+8 IRQ mask bits (Read/Write) 0x00H after reset

This register holds the mask bits for the hardware interrupt selection. To enable an interrupt request line you should write a "1" to the specific interrupt mask bit. For example to enable IRQ5 you should write 0x01 to this address

BIT 0 - IRQ5 selection bit
BIT 1 - IRQ6 selection bit
BIT 2 - IRQ7 selection bit
BIT 3 - IRQ10 selection bit (Only available on DM6852)

BA+9 IRQ mask bits (Read/Write) 0x00H after reset

BIT 0 - IRQ11 selection bit (Only available on DM6852)
BIT 1 - IRQ12 selection bit (Only available on DM6852)
All other bits are reserved for future use.

Programming the DM6852HR

This section gives you some general information about programming the DM6852HR board, and then walks you through the major programming functions of the DM6852HR. These descriptions will help you as you use the example programs and function libraries included with this board. All of the program descriptions use decimal values unless otherwise specified.

The DM6852HR is programmed by writing to and reading from the correct I/O-port addresses of the board. These I/O ports were described in the previous section of this chapter. The following example shows how to perform a 8-bit read and write I/O port addresses using "C"-syntax and assembly code:

	Read:	**Write:**
"C"-syntax	var = **inp**(address);	**outp**(address,data);
Assembly	**mov** dx,address **in** ax,dx	**mov** dx,address **mov** ax,data **out** dx,ax

Clearing and setting bits in an I/O port

When you clear or set bits in an I/O port you must be careful not to alter the status of other bits. You can preserve the status of all the bits you do not wish to change by proper use of the bitwise AND- and OR-operators. Using the AND- / OR- operators, single or multiple bits can easily be set or cleared in one line operations.

1) To clear a single bit in a port , AND the current value of the port with the value "B" , where B = 255-2(exp) bit.

2) To set a single bit in a port, OR the current value of the port with the value "B", where B = 2(exp) bit.

Bits are numbered from 0-7 for the low byte of a word and from 8-15 for the high byte of a word. Setting and clearing of multiple bits in a byte or word is more complex.

3) To clear multiple bits in a port, AND the current value of the port with the value "B", where B = 255-(the sum of the values of the bits to be cleared). Note that the bits do not have to be consecutive.

4) To set multiple bits in a port, OR the current value of the port with the value "B", where B = (sum of the individual bits to be set).

Initializing the 8255 PPI

Before you can operate the non isolated general purpose digital I/O through the 8255 PPI it must be initialized. This step must be performed every time you start up, reset or reboot your computer.

The 8255 PPI is initialized by writing the appropriate control byte to the I/O Port **BA+7** The contents of your control word will vary, depending on how you want to configure your I/O lines. Use the control word description in the previous I/O map section to help you program the right value.

Example: Port A -> Output
Port B -> Output
Port C Upper -> Input
Port C Lower -> Output

Control Byte : 88h

Isolated output programming

The optoisolated MOSFET outputs are controlled with a double register structure. The first register holds the next output state to be transferred to the outputs either from a software command of an external trigger. External triggering is discussed in the next passages.

The outputs can be commanded in the following ways, examples are in "C" syntax:

1. *Software controlled byte write*

```
outp(BA, data);          // Loads storage latch
// User code or operations may be located here
outp(BA+1, anything) // Transfers storage latch contents to outputs
```

2. *Software controlled direct word write*

```
outpw(BA, data);
// Loads storage latch and transfers storage latch contents to outputs
```

3. *Externally triggered output*

```
// Program external trigger and interrupt conditions here
outp(BA, data);
// External trigger will transfer data to output latch
```

Triggering of isolated digital outputs

As described in the previous section an external trigger pulse may be used to latch the data into the output register. Before you can use an external trigger you must first program your board to enable correct operation. The following steps must be taken.

1. Connect the external trigger signal to pin #2 on the 50-pin expansion connector.
2. Determine which transition should cause the triggering, 0->1 or 1->0 change.
3. Program the Setup Register of your DM6852HR, in address BA+2

```
* If you wish to assert a host interrupt on triggering set bit# 0 to 1
* Set the trigger polarity by bit# 1, rising edge triggering set to 1
                                     falling edge triggering set to 0
* Enable external  triggering with bit# 2, to enable set this bit to 1
```

Example in "C" syntax:

```
#define enable_interrupt     1      //      0  0  1
#define enable_trigger       2      //      0  1  0
#define trigger_polarity     4      //      1  0  0

int read_back;

read_back = inp(BA+2);
outp(BA+2,read_back || enable_interrupt || trigger_polarity || enable_trigger);
// this enables interrupting, sets the edge for rising and enables triggering);
outp(BA+8,0x01);      // Enable hardware interrupt line IRQ5
```

Interrupts

- What is an interrupt ?

An interrupt is an event that causes the processor in your computer to temporarily halt its current process and execute another routine. Upon completion of the new routine, control is returned to the original routine at the point where its execution was interrupted.

Interrupts are a very flexible way of dealing with asynchronous events. Keyboard activity is a good example; your computer cannot predict when you might press a key and it would be a waste of processor time to do nothing while waiting for a keystroke to occur. Thus the interrupt scheme is used and the processor proceeds with other tasks. Then when a keystroke occurs , the keyboard 'interrupts' the processor , and the processor acquires the keyboard data , places it into the memory, and then returns to what it was doing before the interrupt occurred. Other common devices that use interrupts are network boards, A/D boards, serial ports etc.

Your DM6852HR can interrupt the main processor when an external trigger event occurs if interrupts are enabled on the DM6852HR board. By using interrupts you can write powerful code to deal with real world events.

-Interrupt request lines

To allow different peripheral devices to generate interrupts on the same computer, the PC XT bus has eight interrupt request channels (IRQ's). A rising edge transition on one of these lines will be latched into the interrupt controller. The interrupt controller checks to see if the interrupts are to be acknowledged from that IRQ and, if another interrupt is being processed, it decides if the new request should supercede the one in progress or if it has to wait until the one in progress is done. The priority level of the interrupt is determined by the number of the IRQ; IRQ0 has the highest priority IRQ7 the lowest. Many of the IRQ's are used by the standard system resources. IRQ0 is dedicated for the internal timer, IRQ1 is dedicated to the keyboard input, IRQ3 for serial port COM2 and IRQ4 for serial port COM1. Often interrupts 2,5 and 7 are free for the user.

- 8259 Programmable Interrupt Controller

The chip responsible for handling interrupt requests in a PC is the 8259 Interrupt Controller. To use interrupts you will need to know how to read and set the 8259's internal interrupt mask register (IMR) and how to send the end-of-interrupt (EOI) command to acknowledge the 8259 interrupt controller.

-Interrupt Mask Register (IMR)

Each bit in the interrupt mask register (IMR) contains the mask status of the interrupt line. If a bit is set (equal to 1), then the corresponding IRQ is masked, and it will not generate an interrupt. If a bit is cleared (equal to 0), then the corresponding IRQ is not masked, and it can generate an interrupt. The interrupt mask register is programmed through **port 21h**.

-End-of-Interrupt (EOI) Command

After an interrupt service routine is complete, the 8259 Interrupt Controller must be acknowledged by **writing the value 20h to port 20h**.

-What exactly happens when an interrupt occurs?

Understanding the sequence of events when an interrupt is triggered is necessary to correctly write interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the DM6852HR), the interrupt controller checks to see if interrupts are enabled for that IRQ, and then checks to see if other interrupts are active or requested and determines which interrupt has priority. The interrupt controller then interrupts the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed onto the system stack, and a new set if CS and IP are loaded from the lowest 1024 bytes of memory. This table is referred to as the interrupt vector table and each entry to this table is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table, the processor starts to execute code from the new Code Segment (CS) and from the new Instruction Pointer (IP). When the interrupt routine is completed the old CS and IP are popped from the system stack and the program execution continues from the point that it was interrupted.

-Using Interrupt in your Program

Adding interrupt support to your program is not as difficult as it may seem especially when programming under DOS. The following discussion will cover programming under DOS. Note, that even the smallest mistake in your interrupt program may cause the computer to hang up and will only restart after a reboot. This can be frustrating and time consuming.

-Writing an Interrupt Service Routine (ISR)

The first step in adding interrupts to your software is to write an interrupt service routine (ISR). This is the routine that will be executed automatically each time an interrupt request occurs for the specified IRQ. An ISR is different from other subroutines or procedures. First, on entrance the processor registers must be pushed onto the stack before anything else! Second, just before exiting the routine , you must clear the interrupt on the DM6852HR by writing to address BA+3, and write the EOI command to the interrupt controller. Finally, when exiting the interrupt routine the processor registers must be popped from the system stack and you must execute the IRET assembly instruction. This instruction pops the CS, IP and processor flags from the system stack. These were pushed onto the stack when entering the ISR.

Most compilers allow you to identify a function as an interrupt type and will automatically add these instructions to your ISR with one exception: most compilers do not automatically add the EOI command to the function, you must do it yourself. Other than this and a few exceptions discussed below, you can write your ISR as any code routine. It can call other functions and procedures in your program and it can access global data. If you are writing your first ISR, we recommend you stick to the basics; just something that enables you to verify you have entered the ISR and executed it successfully. For example: set a flag in your ISR and in your main program check for the flag.

Note: If you choose to write your ISR in in-line Assembly , you must push and pop registers correctly, and exit the routine with the IRET instruction instead of the RET instruction.

There are a few precautions you must consider when writing ISR's. The most important is, **do not use any DOS functions or functions that call DOS functions from an interrupt routine.** DOS is not reentrant; that is, a DOS function cannot call itself. In typical programming, this will not happen because of the way DOS is written. But what about using interrupts? Then, you could have the situation such as this in your program. If DOS function X is being executed when an interrupt occurs and the interrupt routine makes a call to the DOS function X, then function X is essentially being called while active. Such cases will cause the computer to crash. DOS does not support such operation. A general rule is , that do not call any functions that use the screen, read keyboard input and any file I/O routines should not be used in ISR's.

The same problem of reentrancy exists for many floating point emulators as well, meaning you should avoid floating point mathematical operations in your ISR.

Note, that the problem of reentrancy exists, no matter what programming language you use. Even, if you are writing your ISR in Assembly language, DOS and many floating point emulators are not reentrant. Of course, there are ways to avoid this problem, such as those which involve checking if any DOS functions are currently active when your ISR is called, but such solutions are beyond the scope of this manual.

The second major concern when writing ISR's is to make them as short as possible in term of execution time. Spending long times in interrupt service routines may mean that other important interrupts are not serviced. Also, if you spend too long in your ISR, it may be called again before you have exited. This will lead to your computer hanging up and will require a reboot.

Your ISR should have the following structure:

- Push any processor registers used in your ISR. Most C compiler do this automatically
- Put the body of your routine here
- Clear the interrupt bit by writing to address BA+3
- Issue the EOI command to the 8259 by writing 20h to address 20h
- Pop all registers. Most C compilers do this automatically

The following C example shows what the shell of your ISR should be like:

```
/*---------------------------------------------------------------------------
| Function:     new_IRQ_handler
| Inputs:       Nothing
| Returns:      Nothing         - Sets the interrupt flag for the EVENT.
|---------------------------------------------------------------------------*/
void interrupt far new_IRQ_handler(void)
{

        IRQ_flag = 1;           // Indicate to main process interrupt has occurred
        {
        // Your program code should be here
        }
        outp(BA+3,0x00);    // Clears interrupt on DM6852HR
        outp(0x20, 0x20);    /* Acknowledge the interrupt controller. */
}
```

-Saving the Startup Interrupt Mask Register (IMR) and interrupt vector

The next step after writing the ISR is to save the startup state of the interrupt mask register (IMR) and the original interrupt vector you are using. The IMR is located in address **21h**. The interrupt vector you will be using is located in the interrupt vector table which is an array of 4-byte pointers (addresses) and it is locate din the first 1024 bytes of the memory (Segment 0 offset 0). You can read this value directly, but it is a better practice to use DOS function 35h (get interrupt vector) to do this. Most C compilers have a special function available for doing this. The vectors for the hardware interrupts on the XT - bus are vectors 8-15., where IRQ0 uses vector 8 and IRQ7 uses vector 15. Thus if your DM6852HR is using IRQ5 it corresponds to vector number 13.

Before you install your ISR, temporarily mask out the IRQ you will be using. This prevents the IRQ from requesting an interrupt while you are installing and initializing your ISR. To mask the IRQ, read the current IMR at I/O port 21h, and set the bit that corresponds to tout IRQ. The IMR is arranged so that bit 0 is for IRQ0 and bit 7 is for IRQ7. See the paragraph entitled *Interrupt Mask Register (IMR*) earlier in this discussion for help in determining your IRQ's bit. After setting the bit, write the new value to I/O port 21h with the startup IMR saved and the interrupts temporarily disabled, you can assign the interrupt vector to point to your ISR. Again you can overwrite the appropriate entry in the vector table with a direct memory write, but this is not recommended. Instead use the DOS function 25h (Set Interrupt Vector) or, if your compiler provides it, the library routine for setting up interrupt vectors. Remember , that interrupt vector 8 corresponds to IRQ0 , vector 9 for IRQ1 etc.

If you need to program the source of your interrupts, do that next. For example, if you are using the external trigger on the DM6852HR as an interrupt source, program it to do that.

Finally, clear the mask bit for your IRQ in the IMR. This will enable your IRQ.

-Common Interrupt mistakes

- Remember, hardware interrupts are from 8-15, IRQ's are numbered 0-7
- Forgetting to clear the IRQ mask bit in the IMR
- Forgetting to send the EOI command after ISR code. Disables further interrupts.
- Forgetting to set the hardware interrupt line selection correctly

Example on Interrupt vector table setup in C-code:

```
void far _interrupt new_IRQ1_handler(void );        /* ISR function prototype */
#define IRQ1_VECTOR        3                        /* Name for IRQ */
void (interrupt far *old_IRQ1_dispatcher)
        (es,ds,di,si,bp,sp,bx,dx,cx,ax,ip,cs,flags);    /* Variable to store old IRQ_Vector */
void far _interrupt new_IRQ1_handler(void );

/*---------------------------------------------------------------------
| Function:     init_irq_handlers
| Inputs:       Nothing
| Returns:      Nothing
| Purpose:      Set the pointers in the interrupt table to point to
|               our funtions ie. setup for ISR's.
|---------------------------------------------------------------------*/
void init_irq_handlers(void)
{
        _disable();
                old_IRQ1_handler = _dos_getvect(IRQ1_VECTOR + 8);
                _dos_setvect(IRQ1_VECTOR + 8, new_IRQ1_handler);
                Gi_old_mask = inp(0x21);
                outp(0x21,Gi_old_mask & ~(1 << IRQ1_VECTOR));
        _enable();
}
```

```
|/*---------------------------------------------------------------------
| Function:     restore do this before exiting program
| Inputs:       Nothing
| Returns:      Nothing
| Purpose:      Restore interrupt vector table.
|---------------------------------------------------------------------*/
void restore(void)
{
        /* Restore the old vectors */

        _disable();

                _dos_setvect(IRQ1_VECTOR + 8, old_IRQ1_handler);
                outp(0x21,Gi_old_mask);

        _enable();

}
```

APPENDIX A

==========

DM6852HR/DM6852HR Specifications

Host Interface

Jumper-selectable base address, I/O mapped	
Software selectable interrupts	**5,6,7, 10,11,12**

Digital I/O (non isolated)

Number of lines	**24**
Logic compatibility	**TTL/CMOS**
(Pullup and Pulldown resistors are available)	

Digital Outputs (isolated)

Number of lines	**8**
Isolation Voltage	**1.500V Rms**
Output stage	**N-Channel MOSFET**
	60V Vds
	20A Id (peak, max)
(> 7,5V Supply required for drivers)	

Digital Inputs (isolated)

Number of lines	**4**
Input ranges (jumper selectable)	**3**
Triggering Voltages +5V range	**3,5V**
+12V range	**8,5V**
+24V range	**20V**
Isolation Voltage	**1.500V Rms**

Connectors

Isolated Outputs	**Screw type 24 terminal connector**
	Optional header connector 48-Pin
Isolated Inputs	**Screw type 8 terminal connector**
	Optional header connector 16-Pin
Non Isolated I/O	**Header connector 50-Pin**

Electrical

Operating voltage	**+5V+-8% , 125mA**

NOTES:

(c) RTD Finland Oy 1996-2001 DOC: DM6852.SAM